Initiation aux langages du Web :

JavaScript

Table des matières

1. Introduction

Le JavaScript est un langage de script incorporé dans un document HTML. Il permet d'apporter certaines améliorations au langage HTML comme la possibilité d'exécuter des commandes du côté client ou de créer des animations.

De ce fait, le JavaScript est dépendant du navigateur utilisé et tous les effets, animations n'auront pas le même rendu.

Ci-dessous un exemple de script qui permet d'afficher un pop-up :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
	<title>JS basic</title>
	<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
	<meta http-equiv="Content-Language" content="fr" />
	<script type="text/javascript">
		var message = "Bonjour";
		function coucou(){
			alert(message);
		}
	</script>
</head>
	<body>
		<button type="button" onClick="coucou();">Un bonjour</button>
	</body>
</html>
```

Exemple 1: JavaScript basique

On remarque que :

- les balises « script » sont situées dans l'en-tête ;
- les fonctions commencent par le mot clé « function » ;
- les variables sont déclarées à l'aide du mot clé « var » ;
- les appels se font de manière événementielle, ici grâce à « onClick » ;
- l'appel d'une fonction se fait par son nom auquel on ajoute des « () » ;
- chaque ligne se termine par « ; ».

La fonction « alert » utilisée précédemment permet d'afficher un message. Il existe une multitude de fonctions et il est intéressant de consulter le site ci-dessous pour plus de renseignements :

www.tutorialspoint.com/javascript/index.htm

2. Insertion de JavaScript

Il existe, comme pour le *CSS* plusieurs emplacements où il est possible d'écrire du JavaScript:

- l'en-tête de la page ;
- un fichier externe ;
- les balises ;

a) En-tête et fichier externe

Les balises « script » permettent d'insérer du JavaScript ou de faire référence à un fichier contenant du JavaScript.

```
<script type="text/javascript">
        var message = "Bonjour";

        function coucou(){
                alert(message);
        }
</script>
<script type="text/javascript" src="script.js" />
```

Exemple 2: Insertion JavaScript et référence à un fichier externe

b) Balises

Le code JavaScript contenu dans les balises est toujours attaché à un événement. Les événements sont des attributs des balises HTML et ne sont pas applicables à toutes les balises. Par exemple, une image (« img ») ne pourra pas réagir à l'événement « onClick ».

Les événements

Ci-dessous un tableau récapitulatif des événements :

Événement	Description	Balise HTML	Objet JavaScript
onAbort	L'utilisateur stoppe le chargement d'une image	input, textarea, select, body, frame, frameset	form, window, frame
onBlur	prise du focus	input, textearea, select, body, frame, frameset	form, window, frame
onChange	Changement du contenu	input, textarea, select	form
onClick	clic de la souris	a, body, form, ...	link, document, form
onDblClick	Double-clic de la souris	a, body, form, ...	link, document, form
onError	erreur du script	img, body, frame, frameset	image, window, frame
onFocus	perte du focus	body, frameset, frame, input	window, frame, form
onKeyDown	appuie sur une touche	a, img, body, textarea	link, image, document, form
onKeyPress	appuie sur une touche	a, img, body, textarea	link, image, document, form
onKeyUp	relâche une touche	a, img, body, textarea	link, image, document, form
onLoad	Chargement d'un élément	img, body, frameset, frame	image, window, frame
onMouseDown	clic de la souris	a, body, form, ...	link, document, form
onMouseMove	mouvement de la souris	a, body, form	link, document, form
onMouseOut	sortie de souris	a, body, form, ...	link, area
onMouseOver	survol de la souris	a, body, form, ...	link, area
onMouseUp	relâchement du clic	a, body, form, ...	link, document, form
onMove	déplacement de la fenêtre	body, frameset, frame	window, frame
onReset	réinitialisation	forms	forms
onResize	redimensionnement	body, frameset,frame	window, frame
onSelect	sélection	input, textarea	form
onSubmit	Validation d'un formulaire	form	form
onUnload	Fermeture de la fenêtre	body, frameset, frame	window, frame

3. Les Objets

a) Types primitifs

La caractéristique la plus importante d'un langage de programmation est certainement la faculté de manipuler des variables. JavaScript permet de manipuler les styles suivants :

- les nombres (eg. 1, 12,50, ...) ;
- les chaînes de caractères (eg. "ceci est une chaîne") ;
- les booléens (eg. vrai ou faux).

Il existe également deux types de données triviaux : null et indéfini.

Les variables sont définies avec le mot clé « var » comme montré ci-dessous :

```
<script type="text/javascript">
var pet;
var num_legs, tail;
</script>
```

Exemple 3: Déclaration de variable

Le fait de stocker une valeur dans une variable s'appelle **l'initialisation**.

```
<script type="text/javascript">
var pet;
var num_legs, tail;

pet = "dog";
num_legs = 4;
tail = true;
</script>
```

Exemple 4: Initialisation de variable

Le mot clé « var » est utilisé uniquement lors de la création d'une variable et ne doit plus être utilisé par la suite.

La création et l'initialisation peuvent se faire en même temps.

b) Nommage

Il n'est pas possible d'attribuer n'importe quel nom à une variable et de manière générale il faut respecter les règles suivantes :

- ne pas utiliser les mots réservés ;
- toujours faire commencer le nom par une lettre ou « _ » ;
- JavaScript est sensible à la casse.

Ci-dessous la liste des mots réservés :
boolean, break, byte, case, catch, char, class, const, continue, debugger, default, delete, do, double, else, enum, export, extends, false, final, finally, float, for, function, goto, if, implements, import, in, instanceof, int, interface, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, throws, transient, true, try, typeof, var, void, volatile, while, with.

c) Portée ou périmètre

La portée ou périmètre peut être assimilée à la « région » du programme où la variable est **définie** et **utilisable**. Il existe deux périmètres :

- **global**, cela signifie que la variable est définie partout dans le code JavaScript ;
- **local**, cela signifie que la variable est définie dans une portion de code restreinte, généralement une fonction.

Dans le corps d'une fonction, les variables locales prennent l'ascendance sur les variables globales :

```
<script type="text/javascript">
        var a = 2; // Variable globale
        function add( ) {
                var a = 5; // Variable locale
                var b = 1; // Variable locale
                var c = a + b; // c = 6;
        }
</script>
```

Exemple 5: Notion de périmètre ou portée

d) Les tableaux

Les tableaux permettent de stocker une liste de chaînes de caractères ou de nombres dans une seule variable. On peut créer un tableau de deux façons :

- avec le constructeur

```
var chiens = new Array("caniche", "cocker", "bulldog");
```

Exemple 6: Création de tableau avec constructeur

- par affectation

```
var chiens = ["caniche", "cocker", "bulldog"];
```

Exemple 7: Création de tableau par affectation

Pour accéder aux valeurs stockées, il faut utiliser l'entier de l'index désiré. Ainsi,
« chiens[0] » a pour valeur « caniche », « chiens[1] » a pour valeur « cocker », ...
Pour connaître le nombre d'éléments du tableau il suffit d'utiliser la propriété « length » :

```
chiens.length
```

Exemple 8: Nombre d'éléments d'un tableau

e) Les chaînes de caractères

Les chaînes de caractères permettent de stocker une liste de caractères dans une seule variable. On peut créer une chaîne de caractères de deux façons :

- avec le constructeur

```
var chien = new String("caniche");
```

Exemple 9: Création d'une chaîne de caractères avec constructeur

- par affectation

```
var chien = "caniche";
```

Exemple 10: Création d'une chaîne de caractères par affectation

De la même façon que pour les tableaux, on peut connaître la longueur d'une chaîne de caractères en utilisant l'attribut « length ».

Le caractère « + » sert à concaténer deux chaînes de caractères entres elles :

```
var chien =  "caniche"+" nain";
```

Exemple 11: Concaténation de deux chaînes de caractères

f) Les fonctions

Une fonction est une partie de code réutilisable partout dans la page et qui permet de ne pas dupliquer du code. Un exemple de fonction très utilisée, que nous avons vue précédemment, est « alert ».

Définition

Le mot clé «function » permet de créer une fonction. Ci-dessous le schéma de création général :

```
function nom_de_la_fonction(paramètre1,...,paramètreX){
        instructions
}
```

Exemple 12: Schéma de création d'une fonction

Appel

Pour appeler une fonction il suffit d'utiliser son nom suivi des « () » qui contiendront la liste des paramètres :

```
function bonjour(prenom) {
  alert("Hello "+prenom);
}
bonjour("Magali");
```

Exemple 13: Appel d'une méthode

Dans l'exemple précédent, la fonction ne retourne rien et on l'appelle donc une méthode.

Pour retourner une valeur, il faut utiliser le mot clé « return ».

Valeur de retour

Ci-dessous, la même fonction « bonjour » mais qui, cette fois-ci, retourne une chaîne de caractères :

```
function bonjour(prenom) {
  return "Hello "+prenom;
}
var message = bonjour("Magali");
alert(message);
```

Exemple 14: Appel d'une fonction

g) Les objets

JavaScript est un langage de programmation orienté objet (***OOP***) et permet la création d'instances d'objets.

Création abstraite

L'utilisation du constructeur **générique** « Object » permet d'instancier un nouvel objet :

```
var chien = new Object();
```

Exemple 15: Instanciation d'un objet

Les attributs

Pour ajouter un attribut à un objet, il faut utiliser le caractère « . » :

```
var chien = new Object();
chien.race = "caniche";
chien.poid = 4;
```

Exemple 16: Ajout d'attributs

Pour lire un attribut, il suffit de l'appeler de la même façon :

```
var race = chien.race;
```

Exemple 17: Lecture d'un attribut

Création par constructeur

Il est possible de créer une fonction qui va se charger de l'instanciation de l'objet « chien ». Une telle fonction s'appelle un **constructeur** :

```
<script type="text/javascript">
        function Chien(race, poid){
                this. race = race;
                this. poid  = poid;
        }
...
        var chien = new Chien("caniche", 4);
</script>
```

Exemple 18: Création par constructeur

Définition de fonctions

L'ajout d'une fonction à un objet se fait de la même manière qu'une variable :

```
<script type="text/javascript">
	function aboiement(){
		if(this.poid < 5){
			return "yiyi";
		}
		return "WAOUF";
	}
	function Chien(race, poid){
		this. race = race;
		this. poid  = poid;
		this.aboiement =  aboiement;
	}
...
	var aboiement = Chien.aboiement();
</script>
```

Exemple 19: Appel d'une fonction

4. Les opérateurs

Pour les exemples suivants, nous allons utiliser deux variables a et b qui valent respectivement 6 et 2.

a) Arithmétiques

Ci-dessous un tableau résumant les opérateurs arithmétiques utilisés en JavaScript :

Opérateur	Description	Exemple
+	addition	a + b = 8
-	soustraction	a – b = 4
*	multiplication	a * b = 12
/	division	a / b = 3
%	modulo	a % b = 0
++	incrément	a++ = 7
--	décrément	a-- = 5

b) Comparaisons

Ci-dessous un tableau résumant les opérateurs de comparaisons utilisés en JavaScript, les résultats sont booléens (vrai / faux) :

Opérateur	Description	Exemple
==	égalité	a == b = faux
!=	différent	a != b = vrai
>	supérieur	a > b = vrai
<	inférieur	a < b = faux
>=	supérieur ou égal	a >= b = vrai
<=	inférieur ou égal	a <= b = faux

c) Affectations

Ci-dessous un tableau résumant les opérateurs d'affectation utilisés en JavaScript, nous utiliserons une troisième variable c :

Opérateur	Description	Exemple
=	affectation	c = (a + b) équivaut à c = 8
+=	addition ET affectation	c += a équivaut à c = c + a
-=	soustraction ET affectation	c -= a équivaut à c = c - a
*=	multiplication ET affectation	c *= a équivaut à c = c * a
/=	division ET affectation	c /= a équivaut à c = c / a
%=	modulo ET affectation	c %= a équivaut à c = c % a

d) Logiques

Ci-dessous un tableau résumant les opérateurs de comparaisons utilisées en JavaScript, les résultats sont booléens (vrai / faux) :

Opérateur	Description	Exemple
&&	ET logique	(a && b) = vrai
||	OU logique	(a || b) = vrai
!	Négation logique	!(a && b) = faux

e) Conditionnel

L'opérateur conditionnel « ? » évalue dans un premier temps la condition donnée de manière booléenne puis, en fonction du résultat, choisi entre deux instructions :

- ((a > b) ? 10 : 20) équivaut à 10 ;
- ((a < b) ? 10 : 20) équivaut à 20.

5. Boucles conditionnelles et itératives

a) If

La boucle « if » permet de faire des choix entre deux possibilités ou plus.

If

C'est la forme la plus simple de créer une instruction conditionnelle :

```
if (condition){
        Instructions à exécuter si la condition est vérifiée
}
```

Exemple 20: Boucle if

If / else

Cette boucle permet un contrôle plus fin en autorisant l'exécution d'une instruction si la condition n'est pas vérifiée :

```
var age = 20 ;
if (age > 18){
        alert("Vous êtes apte à conduire !");
}else{
        alert("Patienter encore un peu avant de conduire...");
}
```

Exemple 21: Boucle if / else

If / else if

C'est la forme la plus évoluée de boucle pour créer des expressions conditionnelles. Elle offre une très grande souplesse en permettant la gestion de choix multiples :

```
var age = 20 ;
if (age >= 18){
        alert("Vous êtes apte à conduire une voiture !");
}else if(age >= 16){
        alert("Vous êtes apte à conduire un scooter !");
}else{
        alert("Patienter encore un peu avant de conduire...");
}
```

Exemple 22: Boucle if / else if

b) For

La boucle « for » permet de répéter une instruction. Elle se décompose en trois parties :

```
for (initialisation; condition / test; instruction itérative){
    Instructions
}
```

Exemple 23: Schéma d'une boucle « for »

- L'initialisation permet de mettre en place une variable qui fait office de counter (eg. « var i = 0 ») ;
- la condition où test, s'il échoue, permet d'arrêter l'itération (eg. I < 5) ;
- l'instruction itérative permet d'incrémenter ou décrémenter le compteur (eg. i++) ;

Ci-dessous un exemple de boucle « for » :

```
var chiens = ["caniche", "cocker", "bulldog"];
for(var i=0; i < chiens.length; i++){
        var chien = chiens[i];
        document.write(chien);
}
```

Exemple 24: Boucle « for »

c) For...in

La boucle « for...in » permet d'itérer sur les attributs / propriétés d'un objet. Si on reprend l'exemple précédent, la boucle « for...in » permet d'itérer sur les numéros d'index :

```
var chiens = ["caniche", "cocker", "bulldog"];
for(index in chiens){
        var chien = chiens[index];
        document.write(chien);
}
```

Exemple 25: Boucle « for...in »

Si on applique cette boucle sur l'objet chien, ce sont les attributs de Chien qui s'afficheront : poids, race, …

d) While

La boucle « while » est le moyen le plus simple d'itérer. Elle suit le schéma suivant :

```
while (condition){
    Instructions
}
```

Exemple 26: Schéma de la boucle « while »

Le but d'une telle boucle est de répéter le bloc d'instructions tant que la condition est vérifiée :

```
var chiens = ["caniche", "cocker", "bulldog"];
var count = 0;
while(count < chien.length){
        var chien = chiens[count];
        document.write(chien);
        count++ ;
}
```

Exemple 27: Boucle « while »

e) Do...while

La boucle « do...while » permet de placer la vérification de la condition en fin de boucle, ce qui permet d'exécuter au moins une fois les instructions :

```
do{
    Instructions
}while (condition) ;
```

Exemple 28: Schéma de la boucle « do...while »

Veuillez noter le « ; » en fin de boucle.

6. Contrôle des boucles

JavaScript fournit les outils nécessaires pour contrôler les itérations car il se peut qu'il y ait une situation où il soit nécessaire de quitter une boucle sans avoir atteint la fin de l'itération.

a) Break

Le mot clé break permet de quitter une boucle avant d'avoir atteint la fin de celle-ci. Cela placera le curseur d'exécution du code à la fin de la boucle, c'est à dire après l'accolade fermante « } ».

```
var chiens = ["caniche", "cocker", "bulldog"];
var count = 0;
while(count < chien.length){
        var chien = chiens[count];
        if(chien == "cocker"){
                break;
        }
        document.write(chien);
        count++ ;
}
```

Exemple 29: Utilisation de « break »

b) Continue

Le mot clé « continue » permet de « sauter » une itération en passant directement à la suivante :

```
var chiens = ["caniche", "cocker", "bulldog"];
var count = 0;
while(count < chien.length){
        var chien = chiens[count];
        if(chien == "cocker"){
                continue;
        }
        document.write(chien);
        count++ ;
}
```

Exemple 30: Utilisation de « continue »

7. Accéder à un élément de la page

Pour accéder à un élément de la page il est possible d'y parvenir en utilisant :

- l'attribut « id » de l'élément avec la fonction « getElementById » ;
- l'attribut « name » de l'élément avec la fonction « getElementByName ».

Pour récupérer la valeur d'un champ, il suffit ensuite d'utiliser l'attribut « value » comme montré ci-dessous :

```
...
<script>
	function showAlert() {
		var message = document.getElementById("message").value;
		if(message.length == 0){
			message = "Pas de message!";
		}
		alert(message);
	}
</script>
<head>
	<body>
		Message: <input type="text" id="message"/>
		<input type="button" value="Afficher le message"/>
	</body>
</html>
```

Exemple 31: Récupération d'une valeur d'un élément

8. Les cookies

Le protocole HTTP est utilisé entre le serveur et le navigateur pour communiquer.

L'inconvénient de HTTP est qu'il ne mémorise pas les états, on dit qu'il est « ***stateless*** ».

Les cookies sont là pour mémoriser des informations et permettre de garder le fil tout au long de la navigation sur un site.

a) Attributs

Un cookie est un enregistrement texte de cinq variables :

- « Expires » contient la date à laquelle le cookie expire ; si laissé vide, le cookie expirera à la fin de la visite du site ;
- « Domain » contient le domaine du site ;
- « Path » contient le chemin de la page qui à créée le cookie ; si laissé vide, le cookie pourra être récupéré de n'importe quelle page ;
- « Secure » si cet attribut contient la valeur « secure » alors le cookie pourra être récupéré uniquement depuis un serveur sécurisé ;
- « Name=Value,... » permet de stocker une suite de clés / valeurs séparées par des « , ».

b) Stockage

La manière la plus simple de stocker des cookies est de manipuler l'attribut correspondant de l'élément « document » :

```
document.cookie = "race=caniche";
```

Exemple 32: Création d'un cookie

c) Lecture

Les cookies sont stockés à la suite et séparés par des « ; ». Il va falloir les séparer et ensuite récupérer leurs attributs :

```
function readCookie()
{
	var cookies = document.cookie; /* Tous les cookies */
	// On va séparer les cookies pour les mettre dans un tableau
	tab_cookies  = cookies.split(';');
	// Pour chacune des valeurs du tableau on sépare l'attribut de sa valeur
	for(index in tab_cookies){
		explode = tab_cookie[index].split('=');
		name = explode[0];
		value = explode[1];
		alert("Clé : " + name + " et valeur : " + value);
	}
}
```

Exemple 33: Fonction permettant la lecture des cookies

d) Effacement

Pour effacer un cookie il suffit de positionner la valeur de son attribut « Expire » à une date dans le passé.

```
function deleteCookie()
{
        var now = new Date();
        now.setMonth( now.getMonth() - 1 );
        document.cookie = "race=;"expires=" + now.toUTCString() + ";" ;
}
```

Exemple 34: Fonction permettant l'effacement des cookies

9. Boite de dialogue

a) Alert

La boite de dialogue « alert » est la plus simple à utiliser :

```
alert("Bonjour");
```

Exemple 35: Boite de dialogue « alert »

b) Boite de confirmation

La boite de confirmation permet de demander le consentement de l'utilisateur :

```
var retVal = confirm("Voulez-vous continuer?");
if( retVal == true ){
        alert("On continue!");
        return true;
}else{
        alert("On arrête!");
        return false;
}
```

Exemple 36: Boite de confirmation

c) Boite de renseignement

La boîte de renseignement permet de faire apparaître un texte avec une zone de saisie :

```
var retVal = prompt("Entez une race de chien : ", "une race ici");
alert("Vous avez entrez: " +  retVal );
```

Exemple 37: Boite de renseignement

10. Gestion des fenêtres

a) Les redirections

Les redirections se font en positionnant l'attribut « location » de l'élément « window » :

```
window.location="https://www.tala-informatique.fr";
```

Exemple 38: Redirection

b) Ouverture

Il est possible d'ouvrir une page en utilisant la fonction « open » de l'élément « window ». Cette fonction comporte trois paramètres :

- « URL » précise l'URL de la fenêtre à ouvrir ;
- « nom » précise le nom de la fenêtre, utilisable pour l'atteindre ;
- « fonctionnalités » précise les fonctionnalités de la fenêtre comme le contenu de la barre de statut, de la barre d'adresse,...

```
window.open ("https://www.tala-informatique.fr","fenetre_tala");
```

Exemple 39: Ouverture d'une nouvelle fenêtre

Ci-dessous une liste des fonctionnalités qu'il est possible d'activer / désactiver :

Fonctionnalité	Description	Valeurs
status	Affiche la barre de statut en bas de la page	1 / 0
toolbar	Affiche la barre d'outil du navigateur	1 / 0
location	Affiche la barre d'adresse du navigateur	1 / 0
menubar	Affiche la barre de menu de la fenêtre	1 / 0
directories	Affiche les boutons standards du navigateur	1 / 0
resizable	Autorise ou interdit le redimensionnement de la fenêtre	1 / 0
scrollbars	Active les barres de défilement	1 / 0
height	Spécifie la hauteur de la fenêtre	chiffre
width	Spécifie la largeur de la fenêtre	chiffre

c) Positionnement

Une fois la fenêtre ouverte, il est possible de la positionner à l'endroit souhaité grâce à la fonction « moveTo » :

```
win_tala = window.open ("https://www.tala-informatique.fr","fenetre_tala");
win_tala.moveTo(0,0) ;
```

Exemple 40: Déplacement d'une fenêtre

d) Fermeture

La fermeture d'une fenêtre se fait grâce à la fonction « close » de l'élément « window » :

```
win = window.open("","win");
win.close() ;
```

Exemple 41: Fermeture d'une fenêtre

Le mot clé « self » permet de fermer la fenêtre courante.

11. Index des exemples

Index des exemples